Global**Logic**®

# AUTOMATION FRAMEWORK

**Technical Documentation** 

April 2020

www.globallogic.com



## Overview of BDD testing

Behavior-driven development or BDD is a more accessible and effective way for teams new to agile software delivery to test business behavior, rather than a computer function.

BDD approach uses basic concepts of "given, when, then" to describe various user scenarios. It offers an improvement in communication methods between product owners, developers, testers, and users with or without a testing tool. BDD is commonly used with automation using Gherkin and combined with unit testing.

## Advantages of BDD

- Automation engineers no longer define 'test' but are defining 'behavior.'
- Communication of business requirements between developers, testers, and product owners improves.
- The learning curve is much shorter when explaining BDD as it uses simple language.
- Being non-technical, it can reach a wider audience.
- The behavioral approach defines acceptance criteria before development.

BDD helps develop, test, and think about the code from the business owner's point of view.

# Common problems with automation

Here are some of the common problems faced by the Automation QA while automating scripts. These must be addressed at the beginning of the development to avoid inefficiency and code repeatability later on.

## Code Duplication

Most mobile development applications are developed on multiple platforms, commonly on iOS and Android. When developers try to automate application functionality, they need to write the same script multiple times to support all platforms that led to code duplication.

1

EXPLORER			♥ TireList.vue ×			
OPEN EDITORS						
× V TireList.vue components	м		19		<pre>V TireList.vue &gt; {} "TireList.vu await this.\$axios</pre>	
BORAS-BIL-CARS					.post(	
.nuxt			21		process.env.ELASTIC_U	
vscode		32			size: 25,	
assets					from: this.\$route.qu	
components	•	32			query: {	
> Lottie					bool: {	
> skeleton					<pre>must: this.getFi }</pre>	
V Car.vue					},	
CarFilter.vue		330			<pre>sort: this.getSort()</pre>	
Carlnfo.vue		331 332			}	
CarList.vue		333			<pre>, .then(({ data, headers })</pre>	
CarListOpen.vue		334			this.cars = this.cars.c	
CarOpen.vue		335			<pre>\$state.loaded()</pre>	
CarOpenSidebar.vue		336 337			this total and the	
CloseApp.vue		338			<pre>this.\$store.commit('car })</pre>	
Logo.vue						
Post.vue		340		th	<pre>is.cars_loading = false</pre>	
PostInfo.vue PostList.vue		341 342		},		
		343		let	arImages ( <i>car</i> ) { t keys = Object.keys(car	
PostListOpen.vue PostOpen.vue		344		key	<pre>/s = keys.filter(item =&gt; j</pre>	
		345				
PostOpenSidebar.vue PostsFilter.vue		346 347		if	(keys.length === 0) {	
README.md		348		3	eturn false	
lireFilter.vue						
iral interes	_	350		con	st array = []	
outs	M	351 352		key	s.map((item) => {	
ddleware		353		) )	rray.push(carsource.meta	
de_modules		354				
jes				retu	Irn array	
gins		356 357		},		
tic		358		getKey	ByValueImage ( <i>value</i> ) { value) {	
e		359		CO	nst object = this.manufac	
:kerignore		360			return obj.id === value.p	
torconfig		361 362		3)		
ntrc.js		363				
nore		364			(object.length >= 1) {	
ıb-ci.yml		365		}	return object[0].acf.car_n	
oy.sh		366 167		}		
er-compose.production.yml		68				
erfile		69	3,		n false	
fig.json		70			yValueImageLocation (value	
config.js		71				
ige-lock.json		72 73		Cons	t object = this.locations	
ige.json		74		})	eturn obj.id === value[0]	
ME.md				30		
nt.config.js	37			if (	object.length >= 1) {	
ock	37			Ге	turn object[0].name	
	37			8		
	38					
	38			eturn	false	
	38: 38:		},			
PTS	384		cal	Culate	LoadMoreSkeleton () {	
040	385				ert = this care 1	
				(C)p	eof window === 'undefined	
A CONTRACTOR OF THE OWNER OWNER OF THE OWNER OWNE						
The second s		100 100 200	10		the second s	

It also includes duplication of code in debugging tools like taking screenshots, video recording, and logs management.coming from. When an IP spoofing attack occurs, this source details that IP address specifies the sender of the packet is not actual, but a fraudulent IP address which is permitted to access the website. This will make the server handle the request packet as it is coming from the permitted user. Then the server will grant access to the attacker causing various security threats.

## Test Suite

After developing scripts, the developer's second most common challenge is to create the test suite to execute particular scripts as per the client's requirements. In some cases, there is also a need to run scripts based on conditions like:

- Executing specific tests instead of a complete package.
- Executing platform-specific tests, for example, Android-only or iPhone-only.
- Conducting tests with simulators.
- Executing tests dependent on real devices.

### Custom Report

The most important aspect of any action is its result and, in the case of automation, its execution report. However, the most commonly used automation does not provide us with detailed information that can identify the exact cause of the failure.

To overcome these problems, we have developed an Automation framework for accessing automation functionalities, based on the Spock Framework.

2@

3

esc

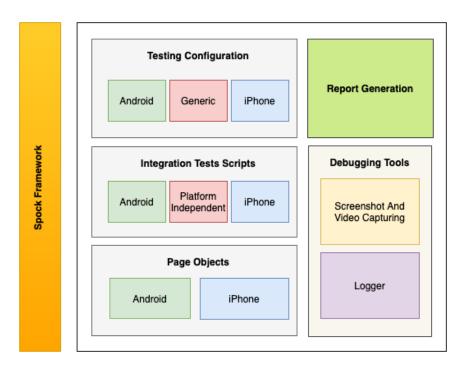


## Overview of the Spock framework

Spock is a test automation framework that uses Groovy to write automation scripts and forces automation developers to follow BDD conventions. The Spock framework also supports parameterized test scenarios that help in testing particular scenarios with different conditions.

## Our Customized Automation Framework

The diagram below gives insight into how the framework works.



This framework aims to provide the initial platform the ability to write integration tests of the multi-platform application. It is based on the following stacks:

- Appium
- Spock framework
- Groovy
- Java



### Components of Framework

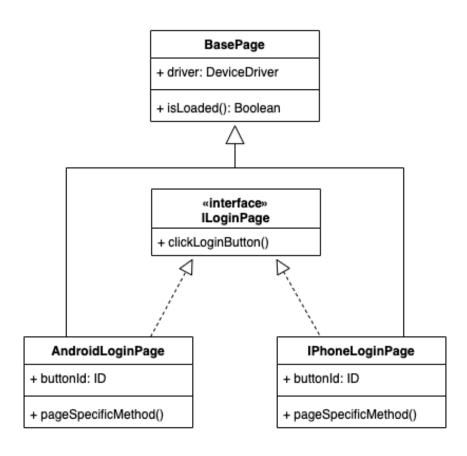
The framework consists of the following major components:

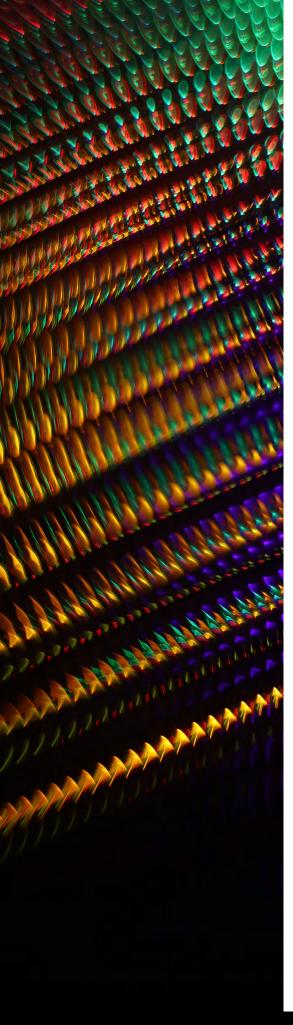
- 1. Page Objects
- 2. Integration Tests
- 3. Custom Debugging Tools
- 4. Reports Generation

#### 1. Page Objects

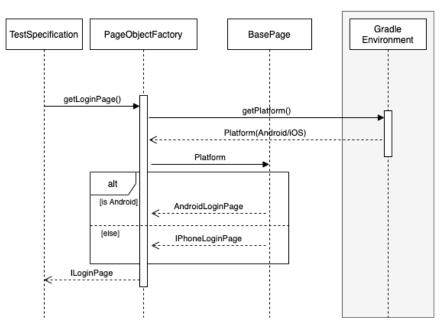
This component provides the application screens in an easily accessible format. The page object is created based on the platform value passed in the Gradle environment. All action methods are accessible only through the standard interface.

Class diagram of Page classes:





Sequence diagram for PageObject creation:



PageObject consists of the following major components:

- Pages(Screen)
- Page Modules
- Other Utilities
- Application.java(App flow)

#### Pages(Screen)

One of the most critical parts of the framework, Pages(Screen) interacts directly with the UI screen and performs all activities related to the user interface on behalf of the user.

#### Page Modules

Page Modules combine the typical set of methods to call to perform a particular sequence in the application. It helps connect processes like entering a username, entering a password, and clicking the Login button for login functionality.

#### Utilities

This contains the utility classes used in different modules such as Constants, Formatters, classes to access property files, and interacting with files to extract necessary information.

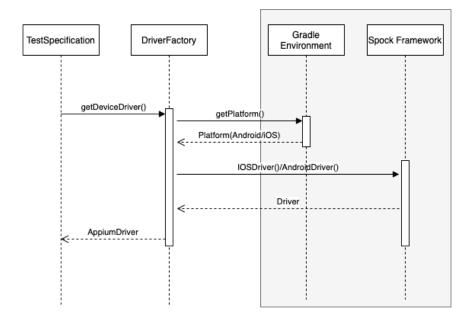
#### Application.java

This handles those functionalities not specific to the particular screen. In other words, it works on the application level to maintain the states. Application.java handles specific scenarios such as common alerts, notifications that can appear on any screen.



#### 2. Integration Tests

This is where automation scripts are written using Gherkin statements. A platform-specific device driver acts as a bridge to interact with the device screen for executing any scripts. The following sequence diagram provides clarity on how the device driver creation happens.



#### 3. Debugging

This component provides debugging tools to investigate test failures. It consists of the following sub-components:

- Screenshots and video capturing
- File logging

#### Screenshots

This provides the ability to capture device screenshots in case of a test failure, so the developer can configure it to capture periodically. Developers can also configure to capture a video recording of each scenario execution. Save captured files in a folder with the test ID name and a filename titled for each scenario.

#### File logging

This provides the ability to capture logs generated during test execution into the file for debugging. Save generated files in their respective test ID folders. The filename here is also the title of each scenario.



#### 4. Reports Generation

This component can generate test reports, customize using CSS, and provide the capability to add details such as:

- Tests count
- Test passed
- Test failed
- Passing percentage
- Platform information
- Scenario details
- Total time taken for execution
- Test ID, which can be linked with the project management tools
- And many more.

## References

#### **Online Resource**

• http://spockframework.org/